LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-500954

GPU-based Scalable Volumetric Reconstruction for Multi-view Stereo

H. Kim, M. Duchaineau, N. Max

September 26, 2011

IVCNZ 2011 : Twenty-sixth International Conference Image and Vision Computing New Zealand
Auckland, New Zealand
November 29, 2011 through December 1, 2011

Disclaimer

GPU-based Scalable Volumetric Reconstruction for Multi-view Stereo

Hyojin Kim
Department of Computer Science
University of California, Davis
Davis, California, USA
Email: mrhkim@ucdavis.edu

Mark Duchaineau
Lawrence Livermore National Laboratory
Livermore, California, USA
Email: duchaineau1@llnl.gov

Nelson Max
Department of Computer Science
University of California, Davis
Davis, California, USA
Email: nlmax@ucdavis.edu

***Abstract*—We present a new scalable volumetric reconstruction algorithm for multi-view stereo using a graphics processing unit (GPU). It is an effectively parallelized GPU algorithm that simultaneously uses a large number of GPU threads, each of which performs voxel carving, in order to integrate depth maps with images from multiple views. Each depth map, triangulated from pair-wise semi-dense correspondences, represents a view-dependent surface of the scene. This algorithm also provides scalability for large-scale scene reconstruction in a high resolution voxel grid by utilizing streaming and parallel computation. The output is a photo-realistic 3D scene model in a volumetric or point-based representation. We demonstrate the effectiveness and the speed of our algorithm with a synthetic scene and real urban/outdoor scenes. Our method can also be integrated with existing multi-view stereo algorithms such as PMVS2 to fill holes or gaps in textureless regions.**

I. Introduction

One of the main approaches for photo-realistic 3D scene reconstruction is image-matching-based multi-view stereo, that uses image correspondences with a geometric relationship, known as epipolar geometry, between pair-wise images [14]. Image correspondence typically refers to dense correspondence algorithms where most or all pixels in the reference image are to be matched, in order to reconstruct a dense geometry of the scene. With computed correspondences and camera poses, final 3D points are obtained via triangulation.

Such a set of 3D points, called a "point cloud", is the final output of most scene reconstruction methods in this category. The point cloud generally does not provide additional information such as normals and connectivity of points. To generate a complete 3D polygonal model, a further process, so-called surface reconstruction, is needed. Since most surface reconstruction algorithms such as [9] take unorganized points, they are useful in many applications; however, since they discard useful underlying information such as normals, obtained from multi-view stereo, they are not appropriate when the surface of an object contains outliers or regions of high curvature. Noise in 3D points due to matching error may also cause incorrect polygonal fitting in surface reconstruction algorithms.

Another research trend in 3D scene reconstruction is object-based volumetric stereo [7]. One of the most widely used methods is space carving [11], [12] that uses photo-consistency, where any photo-inconsistent voxel is carved away. This method assumes that camera poses are known so that every voxel is projected into all images in which it is visible. Since camera poses are typically unknown in most outdoor scene images, the poses should be estimated prior to the reconstruction, which is still a hard problem. Incorrect camera poses can lead to poor reconstruction; therefore volumetric stereo methods may not be appropriate for outdoor scene reconstruction. In addition, they are often "object-centric" where most views look at a centered foreground object, whereas outdoor scene images are more complicated.

In this paper, we propose a novel GPU-parallelized volumetric reconstruction algorithm for aerial photographs of outdoor scenes. We use image-matching-based multi-view stereo to integrate depth maps with images into a high resolution volume. This method is an effectively parallelized GPU algorithm that simultaneously uses a large number of GPU threads, each of which performs voxel carving, given input images from multiple views and their depth maps. Each view's depth map, Delaunay-triangulated from pair-wise stereo correspondences, explicitly represents a view-dependent 3D surface of the scene. The output of this algorithm is a set of surface voxels, each of which stores a weighted average plane among several view-dependent planes that intersect the voxel cube. These surface voxels are suitable for point-based rendering. For large-scale reconstruction in a high resolution voxel grid, we also present a scalable streaming reconstruction scheme along with the GPU-adapted voxel carving. The experimental results show that our method achieves high performance and works effectively for large-scale reconstruction with synthetic and real urban scenes.

This paper is organized as follows: Section II provides an overview of previous work, together with our contribution. Section III describes our proposed GPU-based reconstruction method. Section IV provides some experimental results and discussion, followed by conclusions in section V.

II. Related Work and Contributions

Our approach is among the multi-view integration techniques that reconstruct a scene in a volumetric or polygonal surface representation, given a set of depth maps from stereo correspondences [4]. Such depth map integration methods were originally proposed in reconstruction techniques for range images obtained from a laser scanner. These techniques

can also be used for multi-view stereo applications since each depth map can be treated as a range image.

Curless and Levoy's algorithm [3] is a representative method among the volumetric reconstruction techniques using laser-scanned range images. This paper introduces a weighted signed distance function of each point to the nearest range surface from multiply scanned views to combine structured data. Davis et al. [5] also use the same distance function to fill holes caused by line-of-sight constraints.

Similar reconstruction techniques that directly merge a set of range images into a single mesh have also been proposed. Turk and Levoy [16] describe an incremental algorithm to build surfaces by zippering multiple range meshes, by removing redundant surfaces, clipping the meshes, and zippering along the remaining boundaries. However, difficulties in merging multiple range meshes may be encountered, since there are numerous intersections among triangles [4].

Ju et al.'s algorithm [10] uses a direct mesh integration for multi-view stereo. Zach et al.'s multi-view mesh integration method [17] uses a global energy function to achieve smoothness when noise is present in the depth maps. Bradley et al.'s algorithm [1] merges binocular depth maps using a Delaunay-triangulation-based interpolating mesh approach.

There are several contributions in this paper. First, we present a fast and robust volumetric integration method using graphics hardware to reconstruct an outdoor scene into a volumetric or point-based representation. We believe volumetric integration is more appropriate to transform into parallel GPU code than direct mesh integration. Our proposed method is a GPU-based voxel carving that effectively utilizes massively parallel GPU threads. In addition to adopting some concepts from [3], we combine depth maps with the associated images to generate a set of surface voxels, each with an optimally interpolated color from all visible images and a weighted average plane in a quantized format to efficiently use the GPU memory. For better point-based splatting, the output surface voxels are shifted to a point on the voxel's weighted average plane.

Second, we also focus on scalability of our algorithm to achieve large-scale integration. Due to the limited GPU memory, it is not possible to reconstruct a high-resolution volume in one GPU kernel. Our large-scale reconstruction scheme achieves scalability by partitioning an entire volume into sub-volumes, each of which is reconstructed by the parallel GPU kernel.

In addition, we do not require dense correspondences for depth maps. In dense correspondence algorithms, pixels in textureless regions are difficult to correctly match, which can cause matching errors. Pixels in an occluded region also cause matching errors, and sometimes interpolation and smoothing to correct the matching errors has a negative influence on neighboring correspondences. We believe that semi-dense correspondences are more appropriate than dense correspondences in many scenes. Semi-dense correspondences are sparser matching pairs that are sufficient to determine the entire geometry of the scene. Each depth map, generated from a set of semi-dense correspondences between two images, represents "good" geometry for the scene from this view.

III. Our Reconstruction Algorithm

A. Preprocessing

This algorithm requires a set of input images, each of which comes with its camera pose and depth map. Therefore we first perform camera pose estimation and image matching. In image matching, any dense correspondence algorithm is applicable. For our experiments, we use Cluff et al.'s dense correspondence algorithm [2] using gradient-based warping and coarse-to-fine hierarchical matching (i.e., a coarse matching at a lower resolution level serves as a first approximation for the matching at the next highest resolution level). Given some "good" matching pairs from this algorithm, we extract matching points on or near edges by applying an edge detector to the image. Given camera poses and pair-wise correspondences, we then use these correspondences to compute 3D points, followed by bundle adjustment to refine the camera poses as well as the 3D points.

In some real urban and terrain scenes from aerial photographs, the 3D point cloud makes a large angle with the image plane because of the tilt of the camera. Since these scene models are fairly flat, we rotate these terrain models so that the x and z coordinates are horizontal and the y coordinate is vertical, for efficiency in volumetric representation. Then we can use a much smaller range in the y-axis than in the x- and z-axes, for instance, an $8192 \times 512 \times 8192$ voxel grid. This makes the reconstruction more efficient, with a smaller voxel grid. Given the output stereo correspondences, we first align these points onto the x-z plane by estimating a least squares plane and transforming all coordinates as well as the camera projection matrices.

B. Depth Map Generation

Given semi-dense correspondences between two images, and the 3D points they determine, a depth mesh is generated from a Delaunay triangulation of the 2D correspondences. The 2D vertices in the mesh are replaced with their corresponding 3D points so that the mesh becomes a view-dependent 3D surface. Each triangle's normal is computed from its 3D vertex coordinates, with the sign chosen to make a negative dot product with the viewing direction. If the input correspondences are noisy, mesh simplification algorithms can be applied to merge triangles that have similar normals. Once a depth mesh is prepared, it is projected onto the image plane to create a dense depth map. The reason to use depth maps instead of depth meshes is for fast carving on a voxel-by-voxel basis.

The depth map stores not only depth values, but also plane information with normals at every pixel. Depth values represent a distance between the camera view point and the view-dependent surface. Some view-dependent surfaces may not be correct surfaces, but the incorrect surfaces should be carved away when voxel carving is performed with another view point. On the other hand, some other view-dependent surfaces may represent correct surfaces, depending on the

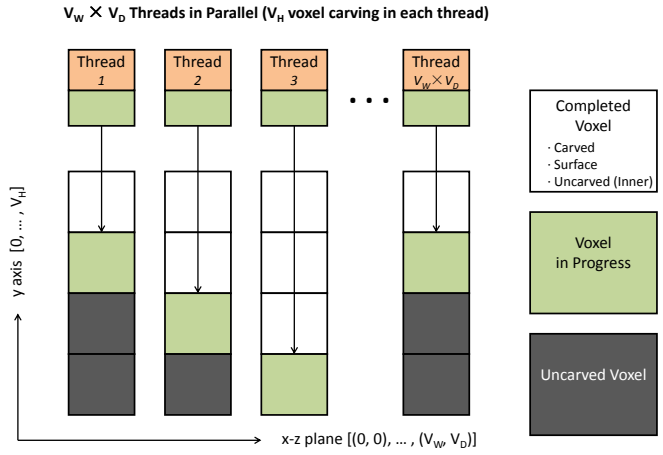


Fig. 1: Parallel voxel carving using GPU threads.

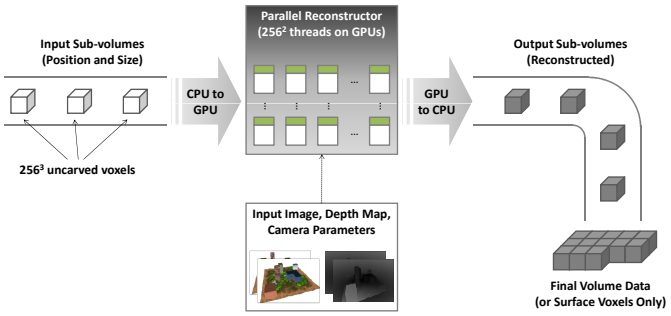


Fig. 2: Scalable volumetric reconstruction scheme.

angle between the camera ray and the object's true surface normal. These surfaces will remain after voxel carving.

C. Parallel Voxel Carving on GPU

This reconstruction method is based on voxel-by-voxel carving, which can easily be transformed into parallel GPU code. Thus voxel carving for a group of voxels is independently executed in parallel as illustrated in Figure 1. Due to the limitation of the GPU memory size, a single reconstruction kernel can handle a volume of approximately 256^3 voxels. The number of GPU threads executed depends on the resolution of the input volume. In the case of 256^3 voxels for instance, 256^2 threads run in parallel and each thread carves 256 voxels independently. For better balancing all thread workloads, voxel carving by each thread is done in the y-axis direction since most urban scenes are fairly flat on the x-z plane.

For each voxel in its y-axis column, the thread loops over the input views and carves away any voxels closer to a particular view than its depth map in its camera projection. Each voxel has a tag that indicates the voxel visibility among three modes: "uncarved", "carved", or "surface." Initially, all voxels are tagged as "uncarved". In each camera view, for each voxel, the distance between the voxel and the camera position is compared with a depth value from the image plane which the voxel is projected onto. If the distance is smaller than the depth value, the voxel is carved. Any carved voxels in a camera projection are no longer examined in later camera projections. If it is larger than the depth, the voxel is preserved. Any uncarved voxel which is close to a depth map within a threshold is tagged as a surface voxel.

For surface voxels, the associated plane information in 4 quantities (normal vector and distance), and an averaged color in 4 components (RGBA) are stored, where the alpha component is used for the voxel mode. To obtain a color from all visible camera views, neighboring colors in each image onto which the voxel is projected are bilinearly interpolated. All the interpolated colors from the multiple views are then averaged using weights depending on the angle between the surface normal and the viewing direction, as proposed by Debevec et al. [6]. Similarly, when multiple planes are intersected within a voxel cube, the normals and distances are weighted averages, just like the signed distance and weight functions in [3]. This computation is independent of the carving order, so it is suitable for parallelization.

The 4 plane quantities are quantized into signed 1-byte values. Normal vector components (A, B, C) and the distance D (i.e., a ratio of the signed distance between the plane and the voxel center to the half-diagonal of the voxel cube) can range from -127 to + 127. All quantized values with -128 indicate no intersection with this voxel and the corresponding plane and those voxels are removed later. As a result, each surface voxel has encoded 8 bytes of plane and color information, which is efficient for the GPU memory.

The output voxels can be rendered using volume rendering or point-based rendering. With the surface voxels, one can use point-based splatting such as QSplat [13] with discs and normals. Each voxel corresponds to a single point which is rendered as a splat (disc) with a small radius. For point-based splatting, the output surface voxels are stored by shifting each voxel's center to a nearest point on the voxel's weighted average plane.

D. Scalable Reconstruction Scheme

One of our goals is to reconstruct large-scale urban scenes in a high resolution voxel grid. As the resolution of the volume increases, overall performance of the volumetric reconstruction drops off rapidly. In addition, each GPU reconstruction kernel can use a maximum 256^3 voxels, since this is all that fits into the GPU memory of our graphics hardware. In this subsection, we present a scalable reconstruction scheme that exploit streaming computation as well as the GPU-accelerated parallel computation. The GPU-based method in Section III. C takes advantage of a number of GPU threads that carve voxels simultaneously. The streaming computation makes it possible to reconstruct large-scale volumes by sequentially processing small regions of the entire volume with the parallelized reconstruction on the GPU.

Figure 2 illustrates an overview of this scheme. Since any subdivision does not affect the final output, this scheme gives scalability; the entire region to be reconstructed can be

Fig. 3: Input images for our experiments. From left to right, *synthetic building*, *Stockton (CA, USA)*, *Walnut Creek (CA, USA)*, and *coffee shack*.

divided into small sub-volumes of a size that fits into the GPU memory. Prior to this process, we determine the range of the entire region by checking the minimum bounding box of the triangulated 3D points. All the generated depth maps, images, and camera projection matrices are stored on the GPU memory before the carving begins, as shown in Figure 2.

In the streaming process, the entire volume is subdivided into sub-volumes and each sub-volume stream has 256^3 voxels. Each sub-volume stores a volume size, position, and resolution, and all voxels in the stream are set to "uncarved". Output sub-volume streams after voxel carving on the GPU contain only surface voxels with color and compressed plane information. Input sub-volume streams in main memory are sequentially transferred into the GPU memory to execute voxel carving accelerated on the GPU. Output carved sub-volume streams are then transferred back to main memory. The main program on the CPU extracts surface voxels in the streams and stores them with the associated voxel position.

IV. Experiments and Discussion

In this section, we evaluate the effectiveness of our algorithm with synthetic and real aerial/outdoor images. Our application was written in C/C++ with CUDA and OpenGL. All experiments were performed on an NVIDIA GeForce 9800 GT with 512 MB video memory, or an NVIDIA NVS 3100M with 512 MB video memory. Figure 3 shows sample input images used for our experiments. For the synthetic scene, we generated an OpenGL textured polygonal model of a building and terrain scene, and captured images from several camera views. For each image, sparse depths (every 2-3 pixels) from OpenGL color/depth buffers were measured to simulate sparse correspondences in real applications. We also used two sets of aerial photographs of real urban scenes; Brett Wayne's aerial images of Walnut Creek, California, USA [20] and Carlos (Kique) Romero's aerial images of Stockton, California, USA [19]. The last scene is our outdoor image set of a coffee shack. For the real scenes, we estimated camera poses using Bundler [15], given intrinsic camera parameters. Pair-wise correspondences to generate depth maps were obtained using Cluff et al.'s dense matching algorithm [2].

A. GPU-based Reconstruction

Figure 4 shows results from our GPU-based single volume reconstruction, both of which have 256^3 voxels. Figure 4 (b) shows the reconstructed volume of the synthetic building scene, given three depth maps. Figure 4 (c) shows the reconstructed volume of the Stockton dataset, given four images with pair-wise correspondences. Since stereo correspondences of this real scene represent a steep slope of the ground with respect to the camera image, we firstly aligned these data onto the x-z plane to efficiently use the volume resolution. Due to the low resolution of the reconstructed volume for the Stockton scene, the result does not provide details. Section IV. B shows improved results by using our large-scale reconstruction scheme, given the same images and correspondences.

TABLE I: Accuracy of the reconstructed synthetic scene. Error indicates a distance between the reconstructed and the ground-truth surface.

# Depth Maps	Min. Error	Max. Error	Mean Error (%)
1	0.0134	35.6	1.5 (0.26%)
2	0.0119	31.4	1.0 (0.17%)
3	0.0122	3.9	0.9 (0.15%)

TABLE II: Performance between CPU and GPU version. Test Set column indicates a test dataset name with its reconstructed voxel grid, the number of images, and the number of depth maps, respectively.

Test Set	CPU Version	GPU Version	Speedup
Synthetic Scene (128^3, 4, 4)	3.3 s	0.08 s	40×
Synthetic Scene (256^3, 4, 4)	19.7 s	0.60 s	33×
Synthetic Scene (256^3, 5, 5)	21.9 s	0.60 s	36×
Stockton Scene (256^3, 4, 2)	13.7 s	0.59 s	23×

Table I describes a quantitative evaluation to measure the accuracy of the reconstructed surface voxels of the synthetic scene, compared to the ground-truth information, using Mesh Lab [18]. As more depth maps are fused, the accuracy increases.

We also evaluated the performance gain of our GPU-accelerated algorithm. For this analysis, we also implemented a CPU version without parallelization. Performance differences between the CPU version and our GPU version are summarized in Table II.

B. Large-scale Reconstruction

We also performed large-scale reconstruction experiments to evaluate the scalability of our algorithm. We used the same Stockton dataset, but applied our streaming and parallel reconstruction scheme. We used $2,560 \times 256 \times 2,560$ voxel

grid, consisting of $10 \times 1 \times 10$ sub-volumes, each of which has a 256^3 voxel grid that fits into the GPU memory. Thus each parallel reconstruction GPU kernel takes a single 256^3 sub-volume and 100 sub-volumes are incrementally reconstructed in the streaming scheme. Then we extracted surface voxels and rendered them using Mesh Lab, as shown in Figure 5 (a). The number of surface voxels are approximately 5.5M.

Another large-scale reconstruction test was performed using the Walnut Creek dataset. In this test, two volume resolutions were tested: $20 \times 1 \times 10$ and $40 \times 1 \times 20$ sub-volumes, each of which has 256^3 voxel grid. Thus the entire reconstructed volumes have $5,120 \times 256 \times 2,560$ and $10,240 \times 256 \times 5,120$ voxels, respectively. The numbers of surface voxels are 4M and 19M, respectively. Figure 5 (b) shows the results using a point-based rendering system, QSplat [13], by converting the reconstructed surface voxels (with normals and actual distances to surface planes) into the QSplat format.

Figure 6 compares two reconstructed results of the Stockton scene: a reconstructed result using Patch-based Multi-View Stereo (PMVS2) [8] and a reconstructed surface voxels using our proposed method. Our result looks more dense and detailed without holes or gaps since our approach effectively integrates multiple depth maps to a single watertight model. For this reason, our method can be used along with other multi-view stereo applications whose result may contain holes or gaps due to matching error in textureless regions (e.g., water in the Stockton scene). Figure 7 shows an example that integrates our proposed method with PMVS2. Figure 7 (a) shows a reconstructed result only using PMVS2 where there are incomplete surface patches due to mismatched keypoints or patch extension failure. Figure 7 (b) shows a result using both PMVS2 and our application where several depth maps generated by PMVS2 are integrated into a single set of surface voxels. The finally reconstructed volume has $1,280 \times 1,024 \times 1,024$ voxels.

V. Conclusion

We presented a new GPU-based volumetric reconstruction algorithm that effectively integrates multi-view images and depth maps. This parallel algorithm uses a large number of GPU threads, each of which independently performs voxel carving and average plane estimation as well as optimal color interpolation. The results showed remarkably improved performance ($20-40\times$ speedup, compared to the CPU-version). Our reconstruction also guarantees scalability using a streaming and parallel scheme for large-scale scene reconstruction. It partitions a high resolution volume into small sub-volumes and performs parallel voxel carving on a GPU. This scheme is straightforward and scalable, depending on volume resolution, GPU memory and performance. The output can be rendered from volumetric or point-based representation. In particular, evenly distributed surface voxels with their normals work effectively for point-based graphics.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

[1] Bradley, D., Boubekeu,T., Berlin, T., Heidrich, W.: Accurate Multi-View Reconstruction Using Robust Binocular Stereo and Surface Meshing. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. Anchorage (2008)

[2] Cluff, S., Morse, B.S., Duchaineau, M., Cohen, J.D.: GPU-accelerated Hierarchical Dense Correspondence for Real-time Aerial Video Processing. In: Proceedings of the 2009 International Conference on Motion and Video Computing, pp. 87–94, Utah (2009)

[3] Curless, B., Levoy, M.: A Volumetric Method for Building Complex Models from Range Images. In: Annual Conference on Computer Graphics - SIGGRAPH, pp. 303–312. New Orleans (1996)

[4] Cyganek, B., Siebert, J. P.: An Introduction to 3D Computer Vision Techniques and Algorithms. John Wiley and Sons (2009)

[5] Davis, J., Marschner, S.R., Garr, M., Levoy, M.: Filling holes in complex surfaces using volumetric diffusion. In: First International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 428–861. Padua, Italy (2002)

[6] Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and Rendering Architecture from Photographs. In: Annual Conference on Computer Graphics - SIGGRAPH, pp. 11–20. New Orleans (1996)

[7] Dyer, C.: Volumetric Scene Reconstruction from Multiple Views. In: Foundations of Image Understanding, pp. 469-489, (2001)

[8] Furukawa Y., Ponce, J.: Accurate, Dense, and Robust Multi-View Stereopsis. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 32, Issue 8, pp. 1362–1376 (2009)

[9] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle,W.: Surface Reconstruction from Unorganized Points. In: Annual Conference on Computer Graphics - SIGGRAPH, pp. 71–78, Chicago (1992)

[10] Ju, X., Boyling, T., Siebert, J.P., McFarlane,N., WU, J., Tillett,R.: Integration of Range Images in a Multi-View Stereo System. In: 17th International Conference on Pattern Recognition, pp. 280–283. IEEE Computer Society Press, Cambridge UK (2004)

[11] Kutulakos, K.N., Seitz, S.M.: Theory of Shape by Space Carving. In: 7th IEEE International Conference on Computer Vision, pp. 307–314. Kerkyra, Greece (1999)

[12] Nitschke, C., Nakazawa,A., Takemura,H.: Real-Time Space Carving Using Graphics Hardware. IEICE Transactions on Information and Systems archive, Volume E90-D, Issue 8, 1175–1184 (2007)

[13] Rusinkiewicz, S., Levoy,M.: QSplat: A Multiresolution Point Rendering System for Large Meshes. In: Annual Conference on Computer Graphics - SIGGRAPH, pp. 343–352. New Orleans (2000)

[14] Seitz, S., Curless, B.,, Diebel, J., Scharstein, D., Szeliski,R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: Conference on Computer Vision and Pattern Recognition, pp. 519–528. New York (2006)

[15] Snavely, N., Seitz, S.M., Szeliski, R.: Photo Tourism: Exploring image collections in 3D. In: Annual Conference on Computer Graphics - SIGGRAPH, Boston(2006)

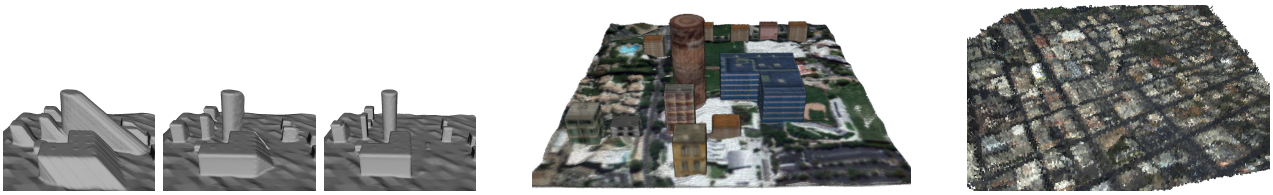
[16] Turk, G., Levoy, M.: Zippered Polygon Meshes from Range Images. In: International Conference on Computer Graphics and Interactive Techniques, Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pp. 311–318. Orlando (1994)

[17] Zach, C., Pock,T., Bischof, H.: A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration. In: IEEE International Conference on Computer Vision - ICCV, pp. 1–8. Rio de Janeiro, Brazil (2007)

[18] Mesh Lab, http://meshlab.sourceforge.net

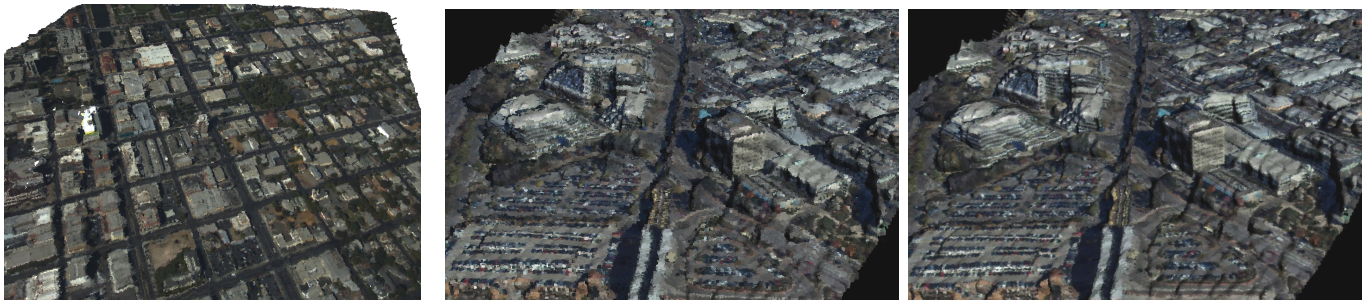
[19] Aerial Images of Stockton, California, (c) Carlos (Kique) Romero, http://www.cognigraph.com/kique_D80-Card1_101NIKON

[20] Aerial Images of Walnut Creek, California, (c) Brett Wayne, http://www.cognigraph.com/walnut_creek_Nov_2005



(a) (b) (c)

Fig. 4: Reconstructed volumes (256^3 voxels). (a) Reconstruction results as more depth maps are fused. (b) Volume rendering of the reconstructed synthetic scene. (c) Reconstructed surface voxels of the Stockton scene, rendered by Mesh Lab.



(a) (b)

Fig. 5: Large-scale urban scene reconstruction. (a) Reconstructed Stockton scene in $2,560 \times 256 \times 2,560$ voxels, rendered by Mesh Lab. (b) Reconstructed Walnut Creek scene in $5,120 \times 256 \times 2,560$ voxels (*left*) and $10,240 \times 256 \times 5,120$ voxels (*right*), rendered by QSplat [13].

(a) (b)

Fig. 6: Comparison with PMVS2 for the Stockton scene. (a) A reconstructed scene using PMVS2 [8]. (b) A reconstructed scene using our proposed method ($2,560 \times 256 \times 2,560$ voxels).

(a) (b)

Fig. 7: Integration with PMVS2 for the coffee shack scene. (a) A reconstructed scene using PMVS2 only. (b) A reconstructed scene using both PMVS2 and our proposed method ($1,280 \times 1,024 \times 1,024$ voxels).